BASEMENT

HD28
.M414
no. 841-76

#340 not used

Dewey

MASS. INST. TECH
MAY 24 1976
LIBRARIES

MASS. INST. TECH.
MAY 21 1976
DEWEY LIBRARY

No. 840 not publ

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Alfred P. Sloan School of Management
Center for Information Systems Research

USE OF VIRTUAL MACHINES IN INFORMATION SYSTEMS

John J. Donovan

REPORT CISR-21
SSWP 841-76

(Supersedes Report CISR-10
SSWP 790-75)

March 1976

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

MASS. INST. TECH.
MAY 21 1976
DEWEY LIBRARY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Alfred P. Sloan School of Management
Center for Information Systems Research

USE OF VIRTUAL MACHINES IN INFORMATION SYSTEMS

John J. Donovan

REPORT CISR-21
SSWP 841-76

(Supersedes Report CISR-10
SSWP 790-75)

March 1976

HD28
.M414
no. 841-76

MAY 24 1976

TABLE OF CONTENTS

ABSTRACT

1. Application Areas Addressed
2. Overview of Proposed System Architecture
3. Analysis of Perfomrance
4. Implicationof Theoretical Results

ACKNOWLEDGEMENTS

REFERENCES

0727381

Abstract

This paper presents a scheme using the virtual machine concept for creating:

1) An environment for increasing the effectiveness of researchers who must use analytical, modeling systems and have complex data management needs.

2) A mechanism for multi-user coordination of access and update to a central data base.

3) A mechanism for creating an environment where several different modeling facilities can access the same data base.

4) A mechanism for creating an environment where several different and potentially incompatible data management systems can all be accessed by the same user models or facilities.

5) A mechanism for reducing the transport cost of integrating existing analytical packages and applications programs under a single unified system.

6) A mechanism for enhancing the data management capabilities of existing modeling and anlaytical languages.

Also presented is a theoretical analysis of the performance implications of this scheme specifically directed at the question of response time degradation as a function of number of virtual machines, of locked time of the data base machine, and of query rate of the modeling machine. A discussion of the practical implications of this anlaysis is given.

Application Areas Addressed

There exists a number of applications that demand a computational facility having interactive data management capabilities as well as having flexible analytical capabilities. The analytical capabilities demanded include facilities for statistical analysis, a broad spectrum of modeling, econometric modeling, dynamic simulation, cross sectional anlaysis, optimization, input/output modleing, graphic presentation, and interpretation of results. These applications include information systems for assisting public policymakers, e.g., analyzing energy problems, as well as the general area of resource management. Other examples include information systems in medical research as well as prototyping of corporate decision support systems. Most present-day computer systems are concerned with operatoinal type problems. For example, computing payrolls, taxes, bills, etc. The class of application above demands a computational facility to support policy decision making. What is different about the problems associated with this class of applications is that:

- problems are not known long in advance;
- problems keep changing; and
- solutions are needed in a short time-frame.

Hence, the research reported here has been to develop a scheme for rapid definition and development of a specific data maangement and analysis structure designed to meet needs as perceived at a particular point in time, but which are known to be evolving. Thus our emphasis is on techniques for accommodating different data base and analysis systems in one integrated framework. Rather than to force the conversion and transport of application systems to one operating system, we advocate the use of

virtual machine concept by which a virtual machine monitor controls the timesharing of the resources of a single physical machine among different operating systems. Netowrk algorithms can be implemented to permit communication of data between the "seemingly incompatible" oeprating systems. Thus with such a system configuration it is possible, for example, to access data from a data management system available under one operating system for use in an analysis program available under another "seemingly incompatible" operating system.

Hence, users of such systems can build upon existing programs and not be required to be retrained in languages as tools they do not presently know.

To fulfill the demands of these applications, we propose using the concepts of simulating many computers onto one computer and have all these simulated computers interconnected to one common virtual machine executing an interactive data management system. Each of the other simulated computers may run a different modeling, simulation, or analytical package even though they may run under incompatible operating systems. Hence, users are not required to learn a new analyatical capability. Each computer may run any existing model or program with no transfer costs.

Such a configuration eliminates the need to devote resources to transporting application langauges and programs between operating systems and permits interaction between application languages and programs not originally envisioned by their developers. For example, an anlaytical package has its datamanagement capabilities greatly enhanced.

Overview of System Architecture

Using this concept, M.I.T.'s Center for Information Systems Research and the M.I.T. Energy Laboratory have developed a system of interconnected simulated computers. The system is called GMIS (Generalized Management Information System) ["GMIS: An Experimental System for Data Management and Analysis", John J. Donovan and Henry D. Jacoby, M.I.T. Energy Laboratory Working Paper No. MIT-EL-75-011WP] and has been applied to a number of energy decision making systems, e.g., the New Enlgand Energy Management Information System (NEEMIS) ["NEEMIS: Text of Governors' Presentation, M.I.T. Sloan School Center For Information Systems Research Report CISR-18]. Much of the software has been developed with the assistance of IBM under an M.I.T./IBM Joint Study Agreement.

It is not the purpose of this paper to describe GMIS. Rather, GMIS is used in this sectionas an operational illustrative example of these concepts.

Currently GMIS is implemented on an IBM System/370 computer. It uses the Virtual Machine (VM) concept extensively.[1] A virtual machine may be defined as a replica of a real computer system simulated by a combination of a Virtual Machine Monitor (VMM) software program and appropriate hardware support. For example, the VM/370 system enables a single IBM System/370 to appear functionally as though it were multiple independent System/370's (i.e., multiple "virtual machines"). Thus, a VMM can make one computer system function as though it were multiple, physically isolated systems.

A configuration of virtual machines used in GMIS is depicted in Figure 1, where each box denotes a separate virtual machine. Those virtual machines across the top of the figure are executing programs that provide user interfaces, whether they be analytical facilities, existing models, or data base systems. All these programs can access data managed by the general data management facility running on the virtual machine depicted in the center of the page. A sample use of this architecture might proceed as follows. A user activates a model, say in the APL/EPLAN machine. That model requests data from the general data base machine (called the Transaction Virtual Machine, or TVM), which responds by passing back the requested data. Note that all the analytical facilities and data base facilities may be incompatible with each other, in that they may run under different operating systems.

---

[1] The VM concept is presented in several places [Parmelee, 1972; Madnick and Donovan, 1974; and Goldberg, 1973], and many of its advantages are articulated elsewhere [Madnick, 1969; Buzen et. al., 1973]. The concept of "virtual machines" has been developed by IBM to the point of a production system release, VM/370 [IBM, 1972].

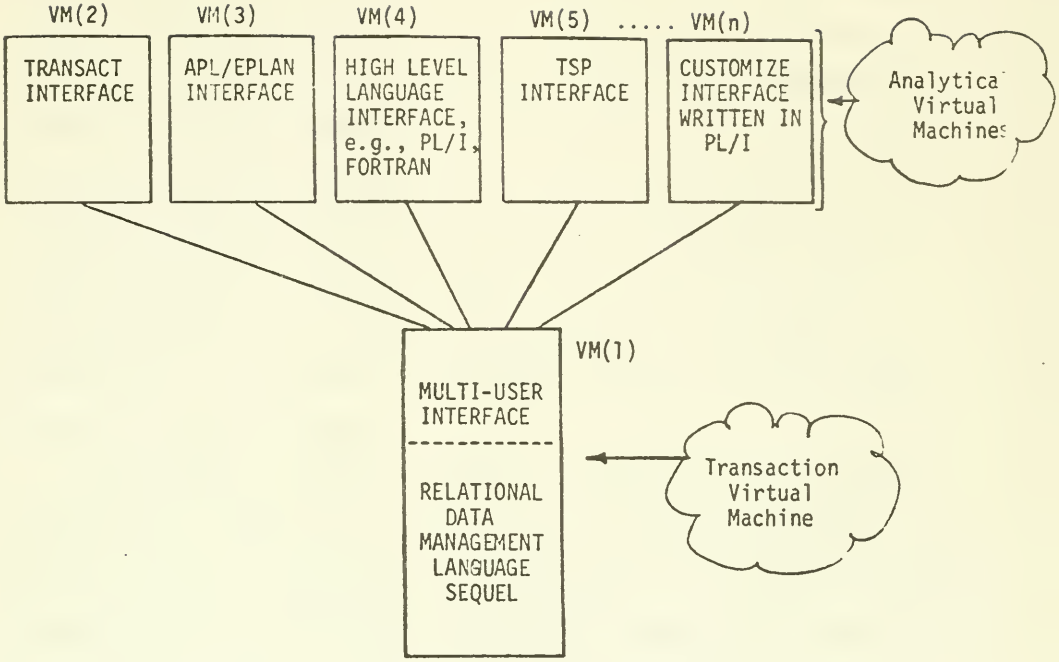

Figure 1: Overview of the Software Architecture of GMIS

The communications facility between virtual machines is incorporated in the program's Multi User Interface. The implementation of this communications facility is described more fully in [Gutentag 75, Donovan and Jacoby 75]. Essentially what is needed is a means of passing commands and data to the data base machine, returning data, and a locking and queueing mechanism. One way to pass data is to use virtual card readers and card punchers. The data base virtual machine would be in wait state trying to read a card from its virtual card reader, the analytical machine would punch the commands on the virtual card reader that would be read by the data base VM. This mechanism is inefficient, however, and does not allow flexible processing algorithms.

The mechanism implemented in GMIS is as follows (note that this mechanism may be invisible to a modeler). Each user virtual machine (UVM), which is accessed by logging on to a separate account ID under VM/370, sends transactions to the Transaction Virtual Machine through a communications facility shared files and virtual card punchers and readers. The Multi User Interface (MUI) stacks these transaction requests and processes them one at a time. The results of each transaction are passed back to the virtual machine that made the request through the same communications facility. Replies to the transactions may be processed with any software interface that is required for the application. Extensions to this architecture to allow interfaces to other data base systems and other computer systems are discussed in [Donovan and Jacoby, 1975].

GMIS software has been designed using a hierarchical approach [Madnick, 1975, 1970; Dijkstra, 1968; Gutentag, 1975]. Several levels of software exist, where each level only calls the levels below it. Each higher level contains increasingly more general functions and requires less user sophis-

tication for use. The transaction virtual machine depicted in Figure 1 shows only two of these levels, the Multi-User Interface and SEQUEL [Chamberlain, 1974]. The data base capabilities of this machine are based on the relational view of data [Codd, 1970].

3. Analysis of Performance

The construction of a system of communicating VM's bring the previously mentioned advantages, but these come at the expense of some sacrifice in performance. Various performance studies of VM's are available in the literature [Hatfield, 1972, Goldberg, 1974], we report here on a theoretical analysis and in the next section of the practical implications of the degradation of variable cost performance as a function of the number of modeling machines. The direction of this work can be seen by considering a configuarion as in Figure 1, where several modeling facilities, each running on a separate virtual machine, are accessing and updating a data base that is managed by a data base management system running on its separate virtual machine. What is the degradation of performance with each additional user? What determines the length of time the data-base machine takes to process a request? What is the best locking strategy?

An access or update to the data-base machine may be initiated either by a user query, which would be passed on by the modeling machine, or by a model executing on the modeling machine. In either case, the data-base machine while processing a request locks out (queues) all other requests. The analysis is further complicated by the fact that as some VM's become locked, then others get more of the real CPU's time, and therefore generate requests faster. However, the data-base VM gets more of

the CPU's time thereby processing requests faster. For example, if there are ten virtual machines, each one receives one-tenth of the real CPU. However, if seven of the ten are in a locked state, then the remaining three receive one-third of the CPU. Thus, these three run (in real time) faster than they did when ten were running.

To try to analyze this circumstance for the uses outlined in this paper, we have assumed that the virtual speeds of VM's are constant and equal. However, when some VM's (including the data-base VM) are allocated a larger share of CPU processing power, they become faster in real time. We assume that each unblocked VM receives the same amount of CPU processing power and at the initial state m machines are running (i.e., the data base machine is stopped if no modeling machines are making requests). 'λ' is the request rate of each modeling VM when there are m VM's running. 'μ' is the service rate at which the data base virtual machine is running when there are m-1 modeling VM and one data base VM running. Thus, we may write the relations:

$$\mu_i = \frac{m}{m-i+1}\,\mu \qquad (i = 1, 2, \ldots, m)$$

$$\lambda_0 = \lambda$$

$$\lambda_i = \frac{m}{m-i+1}\,\lambda \qquad (i = 1, 2, \ldots, m)$$

where i is the number of modeling VM's being blocked. Using a birth/death process model [Drake, 1967], and using a queueing analysis [Little, 1961], we get the following for the response time of the model: where P_i is the steady state probability that there are i modeling machines waiting, and 'N' is the number of modeling machines.

$$T'_{model} = N * \left(\frac{\sum_{i=0}^{m-1} P_i \left(\frac{m-i}{m}\right)}{\sum_{i=0}^{m-1} P_i \left(\frac{m-i}{m}\right) \lambda_i} \right)$$

$$T'_{overhead} = \text{constant}$$

$$T'_{wait\text{-}for\text{-}data} = N * \frac{\sum_{i=1}^{m} iP_i}{\sum_{i=1}^{m} \mu_i P_i}$$

$$T'_{total} = T'_{overhead} + T'_{model} + T'_{wait\text{-}for\text{-}data}$$

4. Implication of Theoretical Results

Figure 2 illustrates the total time to exeucte three different models as a function of the number of modeling VM's. Let us consider some of the implications of the above analysis.

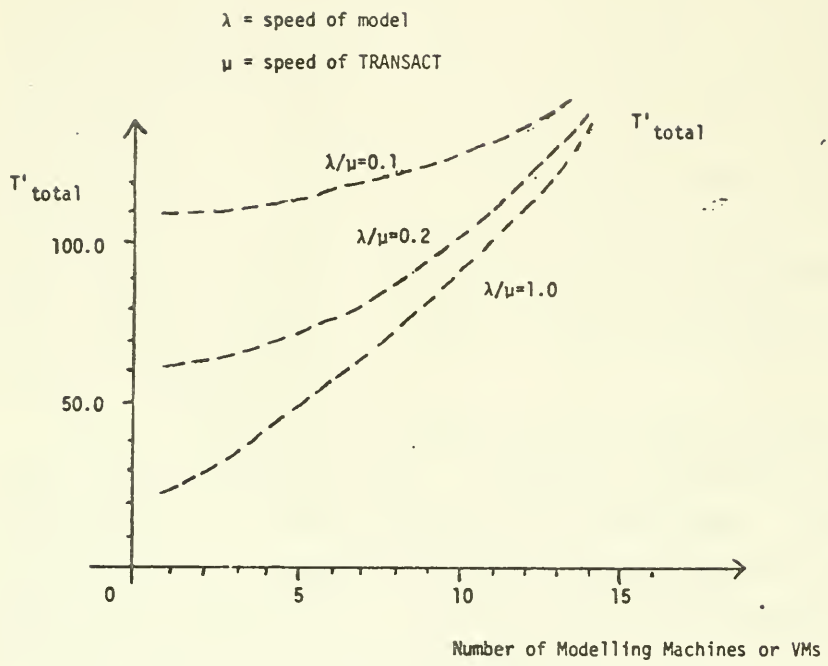

Figure 2. Total Elapsed Times for a VM Configuration

First, for a $\lambda/\mu = .1$, a model executing in a configuration of one modeling machine takes 110 units of time to execute. When the same model, run in an environment of 10 modeling machines all executing similar models, takes approximately 135 units of time to execute -- a degradation of performance of slightly more than 15 percent. Intuitively, λ denotes the speed of the modeling machine, and μ is the speed of the data base machine. Thus a situation where $\lambda/\mu = .1$ indicates that the data base machine is ten times faster than the modeling machine. From the same figure with ratio of $\lambda/\mu = 1$, a model executing with a configuration of one modeling machine takes 20 units of time where with ten machines the same model takes approximately 90 units of time -- over four times longer.

If such a degradation of performance is not tollerable, there are several ways to improve performance. The theoretical study would indicate that increasing μ for a given configuration helps performance. Practically this could be done by changing the processor scheduling algorithm of VM so that the real processor was assigned to the data base management VM more often, thus speeding it up and increasing μ.

Observing the equation for T'_{total} above, another way of reducing T'_{total} is to reduce $T'_{wait_for_data}$. One way to reduce $T'_{wait_for_data}$ is to extend the VM architecture of Figure 1 to allow multiple data base machines. In this configuration $T'_{wait_for_data}$ could be reduced by locking out all data base machines only when one modeling machine is doing a write. For all read requests the multiple data base machines would operate without locking. Shared locks between machines would have to be created as well as a mechanism for keeping a write request pending until all data base machines can be locked.

A way of improving performance further would be to extend the single locking mechanism used in the above multi data base machine configuration to handle multiple locks. Locks would be associated with groupings of data, e.g., a table. The locking policy would be to have all machines only locked out of a portion of the data when one machine was writing into that portion. Thus requests could be processed simultaneously for reads into tables not being written in and for reads to different tables. Thus adding another real processor to the multiple lock VM configuration could greatly improve performance. There is a tradeoff with the multilocking scheme between increases in overhead time in maintaining multiple locks versus increases in wait time for locked data bases. We have not yet extended the theoretical analysis to quantify this tradeoff.

Hence, this study indicates there may be a degradation in performance with multiple users but that there are mechanisms for ameliorating the effects of this degradation.

Other theoretical extensions and analyses of this synchronization model would include extending the model to cover a more common VM operating circumstance -- namely, that where the GMIS system (multiple modeling machines and one data base machine) would have to share the physical machine with other users, also executing under VM, e.g., a payroll program under VS2 under VM, multiple CMS users, etc.

In conclusion, our experience with this approach to date has been very productive. We feel that further studies on cost benefits analysis and on increased effectiveness of users of this sort of system will quantitatively confirm our observation of the benefits of this approach.

Acknowledgement

We wish to acknowledge Professor Peter Chen of M.I.T. for helping with the analytical construction and solutions to the performance equations; Professor Stuart Madnick of Sloan School for helping in formulating these equations; Marvin Essrig's work in applying this VM scheme in developing a system for leading energy indicators and in expanding the scheme for creating an environment where several different and potentially incompatible data management systems can all be accessed by the same models or facility; Drs. Stuart Greenberg and Ray Fessel of the IBM Cambridge Scientific Center for their assistance in implementing the scheme here; Louis Gutentag and the M.I.T. students who worked with him for making the system operational; Jeff Buzen of Harvard for his review of this work; Dave Wood and Professor Henry Jacoby of the M.I.T. Energy Laboratory for their guidance in formulating and articulating some of the ideas expressed here.

REFERENCES

Buzen, J. P., P. Chen, and R. P. Goldberg: "Virtual Machine Techniques for Improving System Reliability," Proceedings of the ACM Workshop on Virtual Computer Systems, March 26-27, 1973.

Chamberlain, D. D. and R. F. Boyce: "SEQUEL: A Structured English Query Language," Proceedings of 1974 ACM/SIGFIDET Workshop, 1974.

Codd, E. F.: "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, vol. 13, no. 6, June 1970, pp. 377-387.

Dijkstra, E.: "T.H.E. Multiprogramming System," Communications of the ACM, May 1968.

Donovan, John J., Jacoby, Henry D.: "GMIS: An Experimental System for Data Management and Analysis," Working Paper No. MIT-EL-75-011WP, September, 1975.

Donovan, John J., Keating, W. Robert: "Text of Gövernors Presentation," Report CISR-18, December, 1975

Drake, A. W.: Fundamentals of Applied Probability Theory, McGraw-Hill, New York, 1967.

Goldberg, R. P.: "Survey of Virtual Machine Research," Computer, vol. 7, no. 6, June 1974, pp. 34-35.

Gutentag, L. M.: "GMIS: Generalized Management Information System -- an Implementation Description," M.S. Thesis, M.I.T. Sloan School of Management, June 1975.

Hatfield, D. J.: "Experiments on Page Size Program Access Patterns, and Virtual Memory Performance," IBM Journal of Research and Development, vol. 16, no. 1, pp. 58-66, January 1972.

IBM: "IBM Virtual Machine Facility/370: Introduction," Form Number GC20-1800, White Plains, New York, July 1972.

Little, J. D. C.: "A Proof of the Queueing Formula: $L = \lambda\omega$," Operations Research 9, 1961, pp. 383-387.

Madnick, S. E.: "INFOPLEX -- Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex," Proceedings of 1975 AFIPS National Computer Conference, 1975.

Madnick, S. E.: "Time-Sharing Systems: Virtual Machine Concept vs. Conventional Approach," Modern Data 2, 3, March 1969, pp. 34-36.

Madnick, S. E. and J. J. Donovan: Operating Systems, McGraw-Hill, New York, 1974.

Morrison, J. E.: "User Program Performance in Virtual Storage Systems," IBM Systems Journal, vol. 12, no. 3, 1973, pp. 34-36.

Parmelee, R. P., T. I. Peterson, C. C. Sullivan, and D. S. Hatfield: "Virtual Storage and Virtual Machine Concepts," IBM Systems Journal, vol. 11, no. 2, 1972, pp. 99-130.